REMARKS

Applicants would like initially to thank the Examiner and supervisor for the courtesy extended on July 28, 2009. The meeting was very helpful and resulted in the amendments to the claims. Claims 2-4, 6-19, 21-33, and 35-49 remain pending. Claims 50-61 are newly added. Claims 2-4, 6-19, 21-33, and 35-49 currently stand rejected under 35 U.S.C. § 103(a).

Amendment

The claims are amended to address concerns raised by the Examiner with regard to the Nachenberg references. Although Applicants disagree generally with the Examiner's interpretation of the Nachenberg '013 reference, an effort is made herein to more clearly distinguish the claimed invention from Nachenberg '013, to bring prosecution to a quick resolution.

The present Application is directed to a technology that *dynamically* verifies code, i.e., as it is executing during its normal use. The newly added claim feature of "continuing execution of the program as long as the verifying determines that the program is valid" is intended to capture elements of "dynamic verification." However, while the method is intended to provide for continued execution "as long as the verifying determines that the program is valid" it is not intended to suggest that any valid program must be continually executed forever in order to read on the claims. Programs may terminate for any number of reasons outside the scope of the methods outlined by the presently amended claims. A distinction between the present claims and Nachenberg '013, however, is that Nachenberg '013 terminates emulation/execution as a part of the virus scan/verification process, whether or not a virus is found in the program.

Additional amendments to the claims were made to correct minor informalities, and provide consistency between dependent claims and claims from which they depend.

Claims 50-61 set forth new method claims and are described in more detail below.

No new matter has been introduced by this Amendment.

Claim Objections

Claim 7 is under objection for containing a minor informality which has been addressed by the present Amendment. Reconsideration and withdrawal of this rejection is therefore respectfully requested.

Claim Rejections - 35 U.S.C. § 103(a)

Claims 2-4, 6-19, 21-28, 30-32, and 35-44 stand rejected under 35 U.S.C. § 103(a) for being unpatentable over U.S. Patent 5,826,013 issued to Nachenberg ("Nachenberg '013") in view of U.S. Patent 6,021,510, also issued to Nachenberg ("Nachenberg '510"). Although not explicitly rejected, the Examiner additionally treated claims 45, 48, and 49 under this heading, and so it is assumed that it was intended to reject these claims under this combination of references. In addition, claims 29, 33, 46, and 47 stand rejected under 35 U.S.C. § 103(a) for being unpatentable over Nachenberg '013, Nachenberg '510, and further in view of the SimOS reference (Office Action, page 15). Applicants respectfully traverse because the prior art fails to teach or suggest each of the features set forth in the claims.

1. Prior art does not teach or suggest "determining an identifying value for a memory block that contains the next instruction"

Each of the independent claims, as a part of the verifying, includes a step for "determining an identifying value for a memory block that contains the next instruction." The Office Action suggests that Nachenberg '013 shows this feature at "figure 4B, 468, three bytes match? – 474, viral signature match." This is incorrect for the following reasons:

First, Nachenberg '013 is not "determining an identifying value for a memory block." Instead, Nachenberg '013 is comparing a list of viral signatures with the contents of a block of memory to determine if there is a match.

Second, Nachenberg '013 is not checking a memory block that contains the next instruction. Instead, Nachenberg '013 is checking memory blocks that were flagged as pages that were written to during the previous emulation phase. The purpose of Nachenberg '013 is to identify polymorphic viruses, which work by executing decryption instructions to decrypt the polymorphic virus. (Refer to Figure 1C of Nachenberg '013, which shows an infected file having a decryption routine 164 for decrypting viral body 160, as described in Nachenberg '013 at col. 5

line 62 to col. 6, line 9.) During emulation phase, Nachenberg '013 tags pages containing instructions that were executed and pages that were altered by the executed instructions. Then, in the scanning phase, Nachenberg '013 scans the tagged pages. See, e.g., Nachenberg '013, col. 9 lines 55-67 and col. 10, lines 13-17. It should therefore be clear that Nachenberg '013 scans pages of previously executed and altered pages, not, "a memory block that contains the next instruction" (emphasis added) as set forth in the claims.

2. The prior art does not teach "ensuring that the program is not executed without dynamically performing the verifying"

Independent claims 2 and 35 now set forth, "ensuring that the program is not executed without [. . .] dynamically performing the verifying." Exemplary methods for implementing this feature are described in the application as filed, e.g., as discussed in sections starting at paragraphs 33 or 75, wherein the verification engine is described.

This further distinguishes the invention from Nachenberg '013 since Nachenberg '013 is not directed to "dynamically performing . . . verifying." Instead, Nachenberg '013 is directed to a method for scanning files for the presence of polymorphic viruses. The emulating/executing occurs with the scanning function to "uncover" any attached polymorphic viruses, but the antivirus scanning system does not have any facility for ensuring that any verification takes place *dynamically* when the program is otherwise executed.

Nachenberg '510 fails to overcome the deficiency of Nachenberg '013. Nachenberg '510 relates to software for maintaining a list of data blocks on a disk (disk sectors) that are known to be free of viruses, so that files formed by the disk blocks can be identified as clean without re-scanning unless the disk sector is modified or the virus signature database is updated. Therefore, Nachenberg '510 provides a performance enhancement to antivirus scanning engines, but does not contemplate performing the check on disk sectors occupied by a currently running program.

3. The prior art does not teach "continuing execution of the program as long as the verifying determines that the program is valid"

Independent claims 2 and 35 now set forth, "continuing execution of the program as long as the verifying determines that the program is valid." This further distinguishes the invention from Nachenberg '013 since Nachenberg '013 terminates emulation/execution of the program

once a file is deemed either "infected" or "not infected" by a virus.

For detecting polymorphic viruses, Nachenberg '013 teaches an emulation phase, which is entered into after failing to detect any static viruses, and a scanning phase, which is entered into (1) periodically (to attempt identification of a partially-decrypted polymorphic virus), (2) when the characteristics are detected that are inconsistent with polymorphic viruses, or (3) when a preselected number of instructions have been emulated (see col. 11, line 66 – col. 12, line 7; [note that "flags have been reset for all polymorphic viruses" means that the file characteristics are inconsistent with all known polymorphic viruses, as explained at col. 7, line 16-20]).

In Nachenberg '013, the file being scanned is deemed "uninfected" when all known polymorphic viruses are excluded, i.e, the decrypted file did not contain a known virus signature or 1.5 million instructions are emulated without revealing a polymorphic virus. See, e.g., column 3, lines 44-46, wherein Nachenberg '013 states, "The emulation control module (220) continues until all mutation engines have been flagged [i.e., the file characteristics are inconsistent with all known polymorphic viruses] or until a threshold number of instructions have been emulated." When the emulation process is terminated for these reasons, "no further emulation of the current target file is necessary. The target file is deemed uninfected . . . and the next target file is loaded . . . for analysis" (col. 12, lines 59-63).

Since Nachenberg '013 terminates the emulation/execution of the program once the file is determined to contain a virus or not contain a virus, the reference does not teach or suggest "continuing execution of the program as long as the verifying determines that the program is valid" as now set forth in each of the independent claims.

Nachenberg '510 fails to overcome the deficiencies of Nachenberg '013 since Nachenberg '510 does not contemplate executing a file at all. Likewise the SimOS reference, which was cited for showing virtualization technology able to switch between direct execution and binary translation (Office Action, page 15) also does not overcome the deficiencies of Nachenberg '013.

4. The prior art fails to teach or suggest ignoring or masking part of the memory block when computing a hash of the memory block

Independent claims 6, and dependent claims 39, 40, and 49 include an operation involving ignoring (or masking) part of a memory block when computing a hash of the memory block. The Office Action cites Nachenberg '013, col. 12, lines 20-30, stating "the selected page locations are scanned and the non-selected ones are ignored" (Office Action, page 6, lines 10-15). Applicants respectfully disagree. Specifically, Nachenberg '013 does not teach generating a hash code from tagged locations of a memory block as suggested in the Office Action. Instead, tagged locations of a ***file*** are scanned to see if ***code segments*** (not ***hash values***) match signatures in the virus database. Nachenberg does not specify that the tagged memory locations be in a common memory block, nor that a ***hash value*** of a memory block is ***ever calculated***. Virus scanning, as described in Nachenberg involves directly comparing code to a signature that is merely a code segment, not a hash value of a code segment. Thus, the "signature" described in Nachenberg is nothing more than a code segment. Even if the "signature" was the same as a hash, there is still nothing in Nachenberg '013 that suggests calculating a hash value from a memory block since Nachenberg '013 only ever compares a memory block with a list of virus signatures.

Claim 7 sets forth, "identifying an indeterminate portion of the current memory block, the indeterminate portion being non-indicative of validity of the current memory block as a whole" and "configuring the mask so that the mask designates at least the indeterminate portion to be ignored when generating the hash value. . . ." The Office Action cites Figure 5 of Nachenberg '013 and specifically element 512 thereof, stating "checks for register modifications" and "exclude all viruses that cannot perform memory write with non-initialized index register" (Office Action, page 7, lines 4-10). Applicants respectfully disagree. Nachenberg '013 describes tracking modifications to memory locations caused by the emulation of virus instructions that decrypt the body of a virus. Each location in memory that is written to therefore potentially contains viral code. Claim 7 specifies that the mask designates that the "non-constant contents . . . be ignored when generating the hash value." Nachenberg '013 in contrast specifically checks these locations in memory. The excluding of viruses that cannot perform memory write described in the Office Action does not refer to excluding locations in memory from scanning, but excluding viruses from the database that the tagged memory locations can potentially match.

I.e., virus scanning involves comparing scanned code with known virus signatures to exclude viruses as being possible matches. If all the signatures are excluded, then the scanned code is considered virus free. See e.g., the discussion of static and dynamic exclusion modules 230, 240 in the paragraph bridging columns 6 and 7 in Nachenberg '013.

5. Dependent claims

The dependent claims, in addition to those noted above, further distinguish the invention from the prior art. For example, with regard to claims 4 and 37, the Office Action cites Nachenberg '013, element 474 (presumably as represented in Figure 4B), stating, "viral signature match determines if the hash value of the instruction is different than all of the reference values" (Office Action, page 5, lines 5-9). However, the viral signature referenced in Nachenberg is not the signature or hash value of a memory block, but are instead virus signatures, which are simply "code segments unique to different viruses" (see Nachenberg, col. 5, lines 43-46).

The above discussion of dependent claims is not exhaustive and merely presents examples of differences between Applicants' claimed invention and cited references. Applicants respectfully request independent reconsideration of the outstanding rejections of the independent claims and each of the dependent claims.

6. New claims

Claims 50-61 make up a new set of method claims that distinguish the invention from the prior art because independent claim 50 specifies "system software" that controls the execution of the instructions being monitored. The term "system software" is defined consistently with the Specification and with the generally understood meaning "system software" among those skilled in the art. In Nachenberg '013, the instructions are not emulated/executed under control of system software. Claim 50 therefore distinguishes the invention from the prior art, and Applicants therefore respectfully submit that Claim 50, and claims 51-61 depending therefrom, should be allowed.

Since the cited prior art fails to teach or suggest each of the features set forth in the claims for at least the above-listed reasons, Applicants respectfully submit that independent claims 2, 6, 35 and 50 are allowable and hence a Notice of Allowance is earnestly and respectfully requested.

Since each of the pending dependent claims include all the features of one of the independent claims, Applicants respectfully submit that the dependent claims are allowable for at least the reasons cited above with regard to the independent claims and further, because they further define and distinguish the invention from the prior art.

Applicants request reconsideration of the outstanding rejections and issuance of a Notice of Allowance. The Examiner is invited to contact the undersigned at 650-427-2390 to discuss any additional changes the Examiner may feel is necessary in light of this Amendment. If any other fees are due in connection with filing this amendment, the Commissioner is also authorized to charge Deposit Account No. 50-2652 (Order No. A043).

Date: October 6, 2009

Respectfully submitted,

/Leonard Heyman/

Leonard E. Heyman
Reg. No. 40,418